

2

AD-A278 706

NPS-OR-94-001



NAVAL POSTGRADUATE SCHOOL

Monterey, California



DTIC
ELECTE
APR 26 1994
S G D

AN ENHANCED DECOMPOSITION ALGORITHM
FOR MULTISTAGE STOCHASTIC
HYDROELECTRIC SCHEDULING

David P. Morton

January 1994

Approved for public release; distribution is unlimited.

Prepared for:
National Research Council
2101 Constitution Avenue
Washington, DC 20418



94-12702

94 114

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5000

Rear Admiral T. A. Mercer
Superintendent

Harrison Shull
Provost

This report was prepared for the Naval Postgraduate School and funded by the National Research Council.

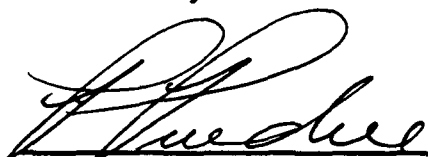
Reproduction of all or part of this report is authorized.

This report was prepared by:




DAVID P. MORTON
NRC Fellow

Reviewed by:



PETER PURDUE
Professor and Chairman
Department of Operations Research

Released by:



PAUL J. MARTO
Dean of Research

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE January 1994		3. REPORT TYPE AND DATES COVERED Technical
4. TITLE AND SUBTITLE An Enhanced Decomposition Algorithm for Multistage Stochastic Hydroelectric Scheduling			5. FUNDING NUMBERS	
6. AUTHOR(S) David P. Morton				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943			8. PERFORMING ORGANIZATION REPORT NUMBER NPS-OR-94-001	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Research Council 2101 Constitution Avenue Washington, DC 20418			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the Department of Defense and the U. S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Handling uncertainty in natural inflow is an important part of a hydroelectric scheduling model. In a stochastic programming formulation, natural inflow may be modeled as a random vector with known distribution, but the size of the resulting mathematical program can be formidable. Decomposition-based algorithms take advantage of special structure and provide an attractive approach to such problems. We develop an enhanced Benders decomposition algorithm for solving multistage stochastic linear programs. The enhancements include warm start basis selection, preliminary cut generation, the multicut procedure, and decision tree traversing strategies. Computational results are presented for a collection of stochastic hydroelectric scheduling problems.				
14. SUBJECT TERMS Stochastic Programming, Hydroelectric Scheduling, Large-Scale Systems			15. NUMBER OF PAGES 28	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

An Enhanced Decomposition Algorithm for Multistage Stochastic Hydroelectric Scheduling

by

David P. Morton

Abstract

Handling uncertainty in natural inflow is an important part of a hydroelectric scheduling model. In a stochastic programming formulation, natural inflow may be modeled as a random vector with known distribution, but the size of the resulting mathematical program can be formidable. Decomposition-based algorithms take advantage of special structure and provide an attractive approach to such problems. We develop an enhanced Benders decomposition algorithm for solving multistage stochastic linear programs. The enhancements include warm start basis selection, preliminary cut generation, the multicut procedure, and decision tree traversing strategies. Computational results are presented for a collection of stochastic hydroelectric scheduling problems.

Keywords: Stochastic Programming, Hydroelectric Scheduling, Large-Scale Systems

January 1994

Department of Operations Research
Naval Postgraduate School
Monterey, CA 93943-5000

Accession For	
NTIS	CRA&I <input checked="" type="checkbox"/>
DTIC	TAB <input type="checkbox"/>
Unannounced <input type="checkbox"/>	
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

1 Introduction

Hydroelectric scheduling is an important planning problem at The Pacific Gas and Electric Company. Depending on hydrological conditions, PG&E's hydroelectric power plants generate roughly 10-20% of the system's annual demand for electric energy. Other energy sources include gas-fired plants, the Diablo Canyon nuclear plant, and imports from external sources; for simplicity, we collectively refer to these as "thermal" energy sources. Thermal energy costing is complex, but for the purposes of the model we describe in this paper, we assume a nonlinear convex thermal cost function. Given a hydro generation schedule, this function provides the cost of operating the thermal system to satisfy the remaining demand for energy. An important source of the thermal cost function's nonlinearity is the different efficiencies of various thermal plants. Hydro units are attractive because they generate energy at a very low variable cost and permit flexible scheduling since they can quickly ramp up to full power. However, due to reservoir and generation capacities and seasonal variations in natural inflow (via precipitation and snowmelt), they cannot be operated at full capacity year-round. The scheduling of the hydroelectric system is further complicated because the volume of future natural inflow into the system's reservoirs is *uncertain*. The objective of the model we describe is to operate the hydro-thermal system with minimum expected cost for a two year planning horizon. Restated: we wish to operate the hydro system so as to maximize expected savings from avoided thermal generation costs. While we give an overview of the hydroelectric scheduling model and coordination (with the thermal system) algorithm in §2, the reader is referred to Jacobs et al. [10] for a more detailed description of the ongoing project at PG&E as well as for references to other approaches to hydroelectric scheduling.

Solutions from hydroelectric scheduling models with deterministic natural inflow forecasts can be unsatisfactory. Hydro generation decisions in such models are made under the incorrect assumption that forecast levels of natural inflow are ensured in forthcoming months; if mean or median inflow values are used, the resulting solutions may fail to hedge against dry inflow scenarios. As a result, in a dry scenario, inefficient and costly thermal plants must be brought on line to satisfy demand for electricity. On the other hand, a conservative strategy derived from a relatively dry forecast may result in forced spills due to finite storage and generation capacities. These spills represent lost

potential energy and lost future cost savings. Stochastic programming formulations allow natural inflow to be modeled as random parameters with known distribution, but the size of the resulting mathematical programs can be formidable. Decomposition or L-shaped algorithms [3,13] provide an attractive approach to such problems because they take advantage of special structure.

The aim of this paper is to present an enhanced decomposition algorithm for multistage stochastic programs and to examine its performance on a set of hydroelectric scheduling problems. The remainder of the paper is organized as follows. In §2 the hydroelectric scheduling module of the stochastic hydro-thermal optimization problem is described; a collection of test problems is also detailed. In §3 we briefly review Benders decomposition algorithm applied to multistage problems and discuss valid cut generation. The empirical performance of several enhancements to the traditional algorithm is presented in §4. In §5 we compare run-times of the enhanced decomposition algorithm and direct linear programming optimizers; the paper is summarized in §6.

2 The Model

A hydrological basin may be viewed as a network consisting of a number of reservoirs (nodes) that are interconnected by rivers, canals, and spillways (arcs); energy is generated as water flows through powerhouses. Given marginal values of energy, we model an individual basin hydroelectric scheduling problem as a T-stage stochastic linear program with recourse (SLP-T):

$$\begin{array}{ll}
 \text{maximize} & \sum_{t=1}^T \sum_{\omega_t \in \Omega_t} p_t^{\omega_t} c_t^{\omega_t} x_t^{\omega_t} \\
 \text{subject to} & -B_{t-1} x_{t-1}^{a(\omega_t)} + A_t x_t^{\omega_t} = b_t^{\omega_t}, \quad 0 \leq x_t^{\omega_t} \leq u_t^{\omega_t}, \quad \omega_t \in \Omega_t \\
 \text{SLP-T for} & t = 1, \dots, T \\
 \text{where} & B_0 \equiv 0.
 \end{array}$$

The sample space for stage t is denoted Ω_t , and a sample point (scenario) in Ω_t is denoted ω_t . A stage $t \geq 2$ scenario, ω_t , has a unique stage $t-1$ ancestor denoted $a(\omega_t)$, and a stage $t < T$ scenario has a set of descendant scenarios denoted $\Delta(\omega_t)$. A_t is an $m_t \times n_t$ matrix and the remaining matrices and vectors are dimensioned to conform. A stage t realization, $\xi_t(\omega_t) = (c_t^{\omega_t}, u_t^{\omega_t}, b_t^{\omega_t})$, is a vector in \Re^{N_t} , where $N_t = 2n_t + m_t$. We assume a finite number of scenarios and a probability mass function given by $P\{\xi_t = \xi_t(\omega_t)\} = p_t^{\omega_t}$. For notational convenience, we have created a first stage sample

space, Ω_1 that is a singleton set where $\xi_1(\omega_1)$ represents the known state at the time decisions are made in the first stage; clearly, $p_1^{\omega_1}$ has value one. SLP-T has $\sum_{i=1}^T m_i |\Omega_i|$ structural constraints and $\sum_{i=1}^T n_i |\Omega_i|$ decision variables where $|\Omega_i|$ denotes the cardinality of Ω_i .

We may, nominally, regard A_i as the node-arc incidence matrix for the hydrological network. The actual form of A_i is more complex for several reasons. First, non-network side constraints must be incorporated; e.g., *decrees* constrain the volume of water in a subset of the reservoirs to a minimum level. Second, subperiod modeling is necessary to capture differences in peak and off-peak values of energy. Third, stages contain different numbers of time periods depending on the corresponding level of uncertainty in natural inflow. For example, summer months are relatively dry and multiple time periods are incorporated in a single stage; the snowmelt season in the spring, on the other hand, is a period of greater uncertainty and shorter time stages are required. In addition, longer time stages are typically used in the later stages of the model.

In SLP-T, decisions occur and uncertainties unfold in the following manner. The first stage hydro generation and storage decisions are made with distributional information on future data; next, a specific scenario is revealed and second stage decisions are made knowing this data, the first stage decision, and conditional probability distributions on future inflows . . . The goal is to operate the hydro basin with maximum expected benefit, in terms of avoided thermal generation cost, for T time stages. The model has essentially three arc types: energy generation arcs, other spatial water transport arcs, and "transition-in-time" arcs. The matrix B_i contains arcs of the third type that are used to equate the amount of water left in a reservoir at the end of one stage with the amount of water in the same reservoir at the beginning of the next stage. Generation, spill, and reservoir capacities appear as simple bounds on the three arc types. Initial reservoir volumes are contained in b_1 ; subsequent $b_i^{\omega_i}$ vectors contain the uncertain natural inflows. Energy generation arcs have objective function coefficients that represent marginal values of energy in terms of dollars per thousand acre foot; these values are stochastic and usually larger in drier natural inflow scenarios. Other arcs typically have objective function coefficients of zero. In general, the stochastic parameters exhibit interstage dependence. For instance, relatively large inflows at lower elevations in the winter months are often coupled with a growing snowpack at higher elevations which will, in turn, lead

to large inflows when melting occurs in the spring. Methods of dealing with finite horizon effects include minimum final period reservoir storage requirements or a final period future value function; both methods may involve scenario dependent constraints.

The individual basin programs described above are subproblems of a larger multistage stochastic *nonlinear* program. Hydro scheduling decisions must be made in a number of basins. The only constraints linking the different basins are the load constraints; these require that at each possible point in time, demand for energy be satisfied. The Dantzig-Wolfe decomposition principle may be applied to create a nonlinear master problem that contains proposed hydro solutions, the demand constraints, and the nonlinear thermal cost function. The linear subproblem separates into a sum of independent subproblems by hydro basin of the form of SLP-T. The Dantzig-Wolfe master generates scenario dependent marginal values of energy for the subproblems and the subproblems, in turn, pass proposed hydro solutions to the master problem. We refer the reader to Eiselt, Pederzoli, and Sandbloom [5] for a discussion of nonlinear Dantzig-Wolfe decomposition.

Name	Size	Dim	Deterministic Equivalent
Moke3.9	169 × 820, 337 × 1713, 673 × 3298	7	7237 × 28473
Ybsf3.9	319 × 1559, 637 × 3119, 1273 × 6239	12	13687 × 53489
Moke4.45	57 × 271, 113 × 548, 337 × 1713, 673 × 3298	7	35736 × 140656
Ybsf4.45	107 × 519, 213 × 1039, 637 × 3119, 1273 × 6239	12	68012 × 265836

Table 1: Test Problems

In §4, we examine the performance of an enhanced decomposition algorithm on a collection of test problems that are preliminary versions of individual basin hydroelectric scheduling problems as described above. The test problems are models with different time horizons, stage definitions, and discretizations of natural inflow distributions. The models are based on two of the larger hydrological basins in the PG&E system: Mokelumne (Moke) and Yuba-Bear-South Feather (Ybsf). In Table 1, "Name" indicates the hydrological basin, the number of stages and number of scenarios; for example, Moke3.9 is a three stage model of the Mokelumne basin with $|\Omega_3| = 9$. "Size" gives the row and column dimensions of the A_i matrices for each stage. The dimension of the domain of the recourse function is the number of large reservoirs in a basin; this value is denoted "Dim." "Deterministic Equivalent" gives the row and column dimensions of the SLP-T formulation.

3 Benders Decomposition and Valid Cut Generation

We may write SLP-2, in minimization form, as follows:

$$\begin{array}{ll} \text{minimize} & \{ c_1 x_1 + E_\omega h(x_1, \omega) \}, \\ \text{subject to} & A_1 x_1 = b_1 \\ & x_1 \geq 0 \end{array} \quad (1)$$

where

$$\begin{array}{ll} h(x_1, \omega) = & \text{minimize} \quad c_2^\omega x_2^\omega \\ & \text{subject to} \quad A_2 x_2^\omega = b_2^\omega + B_1 x_1 \\ & x_2^\omega \geq 0. \end{array} \quad (2)$$

Note the simple upper bounds are not explicit; when this is the case, it may be assumed that they have been included in the structural constraints.

Benders decomposition for SLP-2 (see Van Slyke and Wets [13]) is a resource directed decomposition. A first stage decision is passed to the right-hand-sides of the second stage recourse problems (2) which then act optimally under each scenario ω . Supports of the piecewise linear convex *recourse function*, $E_\omega h(x_1, \omega)$, called *cuts*, are derived from the dual of (2) and are subsequently passed back to the first stage master problem and the process repeats. Under the assumption that second stage “infeasibilities” are modeled by a penalty function, a forward pass of the algorithm generates a feasible decision and hence an upper bound on the optimal objective function value. We can also obtain a lower bound on the optimal objective function value via the master program’s objective function value: the cuts collected so far provide an outer linearization of the recourse function. The extension of this procedure to SLP-T can be viewed as follows. SLP-T is first decomposed into two stages: stage 1 and stage 2, \dots , T . After the first stage problem passes resources to the right-hand-sides of the stage 2 subproblems, there are $|\Omega_2|$ linear programs to solve. Each of these linear programs is solved via decomposition into two stages: stage 2 and stage 3, \dots , T , and so on. This nested method is the “traditional” nested Benders decomposition approach to multistage stochastic linear programming (see Birge [3]); a more formal description is provided in §4.4.

We now provide conditions under which valid cuts can be generated; these results prove useful in developing some of the enhancements to the traditional algorithm described in §4. A *valid* cut is defined to be a cut that lies below the recourse function. Lemmas 1 and 2, state that dual feasible vectors generate valid cuts for SLP-2 and SLP-T, respectively.

Lemma 1 Consider SLP-2 with $|\Omega| = K$. If $(\hat{\pi}^1, \dots, \hat{\pi}^K)$ are dual feasible for the second stage subproblems then these dual vectors generate a valid cut for the first stage master program.

Proof

Let $\Pi^\omega = \{\pi : \pi A_2 \leq c_2^\omega\}$ denote the dual feasible region of (2). By hypothesis, $\hat{\pi}^\omega \in \Pi^\omega \forall \omega \in \Omega$; thus

$$\hat{\pi}^\omega (b_2^\omega + B_1 x_1) \leq h(x_1, \omega) = \max_{\pi^\omega \in \Pi^\omega} \pi^\omega (b_2^\omega + B_1 x_1) \quad \forall x_1.$$

By taking expectations we see $Gx_1 + g$ is a valid cut where $G = E_\omega \hat{\pi}^\omega B_1$ and $g = E_\omega \hat{\pi}^\omega b_2^\omega$. ■

With the exception of the final stage, the difference in the multistage setting is that subproblems generating cuts for their ancestors contain their own cuts. Lemma 1 implies dual feasible vectors to the stage T subproblems generate valid cuts for their stage $T-1$ ancestors, but a new result is needed for the general case. Lemma 2 ensures that if the stage t ($2 \leq t < T$) subproblems contain valid cuts then they will, given dual feasible vectors, generate valid cuts for their stage $t-1$ ancestors. The stage t ($1 \leq t < T$) subproblem under scenario ω_t , denoted $\text{sub}(\omega_t)$, has the following form:

$$\begin{aligned} & \text{minimize} && c_t^{\omega_t} x_t^{\omega_t} + \theta_t^{\omega_t} \\ & \text{subject to} && A_t x_t^{\omega_t} = b_t^{\omega_t} + B_{t-1} x_{t-1} \\ & \text{sub}(\omega_t) && -G_t^{\omega_t} x_t^{\omega_t} + e \theta_t^{\omega_t} \geq g_t^{\omega_t} \\ & && x_t^{\omega_t} \geq 0. \end{aligned} \tag{3}$$

The rows of the matrix $G_t^{\omega_t}$ contain cut gradients; the elements of the vector $g_t^{\omega_t}$ are cut intercepts; and e denotes the vector of all 1's. As the algorithm proceeds, the row dimension of these quantities will grow.

Lemma 2 Suppose $|\Delta(\omega_t)| = K$, $1 \leq t \leq T-2$, and the descendants of $\text{sub}(\omega_t)$ contain valid cuts. If $[(\hat{\pi}_{t+1}^1, \hat{\alpha}_{t+1}^1), \dots, (\hat{\pi}_{t+1}^K, \hat{\alpha}_{t+1}^K)]$ are dual feasible vectors for the descendants of $\text{sub}(\omega_t)$ then these dual vectors generate a valid cut for $\text{sub}(\omega_t)$.

Proof

Denote the value of subproblem (3) by $\mathcal{F}_{t-1}(x_{t-1}, \omega_t, G_t^{\omega_t}, g_t^{\omega_t})$. Let $(\tilde{G}_{t+1}^{\omega_{t+1}}, \tilde{g}_{t+1}^{\omega_{t+1}})$ be a set of valid cuts and define $f_t(x_t, \omega_{t+1}) = \mathcal{F}_t(x_t, \omega_{t+1}, \tilde{G}_{t+1}^{\omega_{t+1}}, \tilde{g}_{t+1}^{\omega_{t+1}})$. For each $\omega_{t+1} \in \Delta(\omega_t)$ there exists a finite set of cuts, denoted by $(\tilde{G}_{t+1}^{\omega_{t+1}}, \tilde{g}_{t+1}^{\omega_{t+1}})$, such that the conditional stage $t+2$ recourse function

is completely specified; let $h_t(x_t, \omega_{t+1}) = \mathcal{F}_t(x_t, \omega_{t+1}, \bar{G}_{t+1}^{\omega_{t+1}}, \bar{g}_{t+1}^{\omega_{t+1}})$. Now suppose $(\hat{\pi}_{t+1}^{\omega_{t+1}}, \hat{\alpha}_{t+1}^{\omega_{t+1}})$ is dual feasible for the program corresponding to $f_t(x_t, \omega_{t+1})$. Then

$$\hat{\pi}_{t+1}^{\omega_{t+1}} (b_{t+1}^{\omega_{t+1}} + B_t x_t) + \hat{\alpha}_{t+1}^{\omega_{t+1}} \bar{g}_{t+1}^{\omega_{t+1}} \leq f_t(x_t, \omega_{t+1}) \leq h_t(x_t, \omega_{t+1}) \quad \forall x_t. \quad (4)$$

The left hand inequality is immediate by writing the program corresponding to $f_t(x_t, \omega_{t+1})$ in its dual form. The right hand inequality follows as the cut constraints in $f_t(x_t, \omega_{t+1})$ are dominated by the cuts of $h_t(x_t, \omega_{t+1})$. The desired result is obtained by multiplying (4) by $p_{t+1}^{\omega_{t+1}|\omega_t}$ and summing over all $\omega_{t+1} \in \Delta(\omega_t)$. ■

4 The Enhanced Decomposition Algorithm

In this section, four enhancements to the traditional nested Benders algorithm are presented: *Warm start* techniques obtain “good” initial basic feasible solutions for subproblems based on optimal basis information from previous subproblem solutions. *Advanced start* procedures generate preliminary cuts prior to initiating a formal Benders algorithm. The *multicut* Benders decomposition algorithm returns one cut for each descendant of a particular subproblem instead of a single aggregate cut. *Tree traversing strategies* prescribe the order in which subproblems of the decision tree are solved.

The enhanced algorithm is implemented in FORTRAN 77, and the computational results we present are from a Hewlett Packard 9000/750 workstation. The algorithm uses NETSIDE, a special purpose code that solves the minimum cost flow network problem *with side constraints*, developed by Kennington and Farhangian as the subproblem solver. NETSIDE is based on a specialization of the primal simplex method (see Kennington and Helgason [11] and Barr et al. [2]); in our setting, the set of side constraints includes Benders cuts.

The performance of a particular algorithmic enhancement will be analyzed with respect to a base case strategy which is the strategy we recommend. Thus, we evaluate the marginal effect of each enhancement. The base case strategy and its performance on the four test problems is summarized in Tables 2 and 3; the details are presented in the respective subsections that follow. All problems are solved to within an objective function tolerance of 0.01%. Reported CPU times exclude input and output operations. The # subproblems column of Table 3 gives the number of subproblems solved in each stage during the course of the algorithm.

<i>Warm Start</i>	Candidate list of length 20 until relative error $\leq 5\%$
<i>Advanced Start</i>	Prespecified Decisions Method with shared cuts
<i>Multicut</i>	Yes
<i>Tree Traversing Strategy</i>	Fastpass

Table 2: Base Case Strategy

Name	# subproblems	# iterations	CPU (sec.)
Moke3.9	10-59-99	10	55.4
Ybsf3.9	7-47-72	7	119.8
Moke4.45	10-64-274-459	10	221.4
Ybsf4.45	8-83-205-369	8	404.0
Average	N/A	8.8	200.2

Table 3: Base Case Strategy Performance

4.1 Warm Start

Similar subproblems are repeatedly solved during the course of a decomposition procedure. Techniques that take advantage of optimal basis information from previous subproblem solutions are critical for developing efficient algorithms. Wets [14] and Garstka and Rutenberg [6] have proposed bunching and sifting techniques, respectively for solving a large number of similar linear programs. The sifting method, however, requires component-wise independence of the right-hand-side and deterministic objective function coefficients; our test problems violate these requirements. Moreover in our experiments there were a low number of "repeat" optimal bases which seemed to indicate bunching might not occur. While the primary computational challenge in the two-stage problem lies in the solution of a large number of similar second stage problems *each iteration*, the greatest potential for computational savings in a multistage problem, with only a few stochastic branches each stage, rests in an ability to select good initial bases for each subproblem.

We propose a warm start technique in which initial bases are selected from a candidate list until the relative error is sufficiently small. In subsequent iterations, subproblems are initialized with the basis from their previous optimal solution. The columns of a basic solution of subproblem (3) can be partitioned into a network component and a side constraint component; see Kennington and Helgason [11]. The heuristic used to select the best basis from the candidate list for a particular

subproblem proceeds as follows:

- (i) Calculate network flows from the *network* component of each candidate list basis.
- (ii) Calculate solutions for the entire subproblem based on each network flow solution.
- (iii) Determine the objective function value of each solution.
- (iv) Select the candidate list basis that has the minimum corresponding objective function value.

Step (i) can be performed quickly due to the tree structure of the network basis. In step (ii) we substitute the solution from (i) into the side constraints and generate a feasible solution from slack, surplus, artificial, and future cost variables. The "best" basis is then determined from the objective function value of each solution. Note that the value calculated in (iii) is only an estimate of the actual objective function value that the basis will yield since the side constraint component of the basis is not considered. We ignore this component due to the computational effort required for refactorization and the fact that the network constitutes most of the subproblem. Minimal storage is required for each basis: arc indices within a pointer structure, a list of upper bound arc indices for the network, and a list of column indices for the side constraints. See Jacobs [9] for the details of the NETSIDE basis insertion procedure. Warm start parameters that the user must select are the maximum size of the candidate lists and the relative error at which the method switches from the candidate list heuristic to simply reusing the previous optimal basis for each subproblem; reasonable values for these parameters are suggested below in the computational results discussion.

Computational Results

Columns 2-4 of Table 4 detail the performance of the algorithm when each subproblem solution begins with an all-artificial basis. Similarly, columns 5-7 show the empirical results of the strategy in which the candidate list heuristic is not used and we simply recall the previous optimal basis for each subproblem; if such a basis does not exist (because a particular subproblem has not yet been solved) then the optimal basis of another subproblem from the same stage is used. The "x increase" column is defined as T/T_{bc} and the "% increase" column as $(T - T_{bc})/T_{bc} \cdot 100$. T_{bc} denotes the running time of the base case strategy and T the modified strategy; e.g., the base case with no warm start. If the % increase column is 20 then it is to be read: "the current strategy's running time is 20% longer than the running time of the base case strategy." Table 4 reveals the dramatic impact

of warm starts and also indicates that substantial computational savings can result from using the heuristic to select good bases early in the algorithm. The values of 5% for the switch over tolerance and 20 for the candidate list length (see Table 2) were determined by varying these parameters on a wider variety of test problems using the base case strategy with and without an advanced start.

Name	No Warm Start			Recall Previous Basis		
	iter.	CPU (sec.)	x increase	iter.	CPU (sec.)	% increase
Moke3.9	13	909.1	16.4	12	75.0	35.4
Ybsf3.9	8	1973.3	16.5	8	163.4	36.4
Moke4.45	11	3793.4	17.1	11	263.4	19.0
Ybsf4.45	11	14267.8	35.3	9	526.1	30.2
Average	10.75	5253.9	21.3	10	257.0	30.3

Table 4: Warm Start Procedures

4.2 Advanced Start

A Benders decomposition algorithm initiated with no preliminary cuts generates myopic first iteration decisions and “extreme” decisions in the early iterations. The idea behind an advanced start method is to calculate preliminary cuts to help guide the early iterations of the algorithm. A technique utilized by Infanger [8] involves first solving, by Benders decomposition, the much smaller *expected value problem* in which the stochastic parameters of SLP-T are replaced by their population means. The cuts produced in the process are valid for SLP-T if the stochastic parameters exhibit interstage independence and appear only in B_t and b_t ; this claim is easily verified via Lemmas 1 and 2. However, in the stochastic hydroelectric scheduling problems the objective function coefficients are random and the stochastic parameters are temporally correlated; thus the expected value method is not applicable and we seek an alternate approach.

A “prespecified decisions” method for computing preliminary cuts requires, for each stage $t < T$, a collection of decision vectors: $\{\hat{x}_t^{(i)} : i = 1, \dots, N_t\}$. The basic idea is to generate preliminary cuts by solving subproblems with right-hand-sides of the form: $B_{t-1}\hat{x}_{t-1}^{(i)} + b_t^{\omega_t}$. A naive implementation involves solving the descendants of each scenario tree node at each of the prespecified decisions and computing the corresponding cuts. In the streamlined approach described in this section, we select a single scenario $\hat{\omega}_t$ on each stage and solve the descendants of scenario $\hat{\omega}_{t-1}$ and then pass a cut

back not only to $\text{sub}(\hat{\omega}_{t-1})$ but also to all its stage $t-1$ neighbors via the *dual sharing formula* described below; Figure 1 summarizes the method. In a symmetric four stage, 45 scenario problem with 1, 3, 15, and 45 scenarios on each stage and $N_t = 3$, the number of subproblems solved in the streamlined method is 33 while the naive method is 189.

```

define  $\hat{\omega}_t, t = 1, \dots, T-1$ 
do  $t = T$  downto 2
  do  $i = 1$  to  $N_{t-1}$ 
    do  $\omega_t \in \Delta(\hat{\omega}_{t-1})$ 
      form RHS of  $\text{sub}(\omega_t)$ :  $B_{t-1}\hat{x}_{t-1}^{(i)} + b_t^{\omega_t}$ 
      solve  $\text{sub}(\omega_t)$ 
    enddo
    pass cut to  $\text{sub}(\omega_{t-1}) \forall \omega_{t-1} \in \Omega_{t-1}$ 
  enddo
enddo

```

Figure 1: Prespecified Decisions Method - Sharing Duals

The only relevant components of the prespecified decision vectors are ones that contribute to the product $B_{t-1}\hat{x}_{t-1}^{(i)}$; this corresponds to end-of-stage reservoir storage volumes in the hydro scheduling problems. Reservoirs have natural lower and upper storage bounds, and in the absence of additional information (e.g., historical storage levels or values from prior optimizations), the prespecified decisions are defined as percentiles between the upper and lower bounds. In particular, we use three prespecified vectors at 20%, 50%, and 80%, and we define $\hat{\omega}_t = \lceil K_t/2 \rceil$ where $\Omega_t = \{1, \dots, K_t\}$ and where the ceiling function $\lceil \cdot \rceil$ gives the smallest integer greater than or equal to its argument. The order of cut computation is relevant; for example, in a three stage problem all preliminary cuts are passed to the second stage prior to passing any cuts to the first stage. In this way, the maximum possible amount of information is subsequently passed to the first stage.

The Dual Sharing Formula

The dual of $\text{sub}(\omega_t)$ (see program (3)) with explicit simple bounds may be written:

$$\begin{aligned}
 & \text{maximize} && \pi_t^{\omega_t} \left(b_t^{\omega_t} + B_{t-1}x_{t-1}^{a(\omega_t)} \right) + \alpha_t^{\omega_t} g_t^{\omega_t} - \mu_t^{\omega_t} u_t^{\omega_t} \\
 & \text{subject to} && \pi_t^{\omega_t} A_t - \alpha_t^{\omega_t} G_t^{\omega_t} - \mu_t^{\omega_t} \leq c_t^{\omega_t} \\
 & && e^T \alpha_t^{\omega_t} = 1 \\
 & && \alpha_t^{\omega_t} \geq 0, \mu_t^{\omega_t} \geq 0.
 \end{aligned}$$

In describing the dual sharing formula, super and subscripts are suppressed for clarity. Suppose we have solved a stage t subproblem with data $(\hat{c}, \hat{G}, \hat{A})$ and obtained optimal dual prices $(\hat{\pi}, \hat{\alpha}, \hat{\mu})$.

Given another stage t subproblem with data (c, G, A) feasible dual prices can be generated directly from $(\bar{\pi}, \bar{\alpha}, \bar{\mu})$ via

$$(\pi, \alpha, \mu) = (\bar{\pi}, \bar{\alpha}, [\bar{\pi}A - \bar{\alpha}G - c]^+). \quad (5)$$

Dual feasibility of (π, α, μ) is easily verified by substitution. The $[v]^+$ notation means take the positive part of the vector v , component-wise. We refer to (5) as the *dual sharing formula*. Note (5) is also valid for stage T subproblems when the (vacuous) cut gradient matrix and associated dual variables are dropped.

Suppose we have solved the descendants of $\text{sub}(\hat{\omega}_t)$; using the corresponding optimal dual variables, the dual sharing formula may be applied to compute a valid cut for $\text{sub}(\omega'_t)$. To compute this cut, we match elements of $\Delta(\hat{\omega}_t)$ and $\Delta(\omega'_t)$. (Another possibility is to select the dual vectors that produce the strongest cut at a particular stage t decision.) By Lemmas 1 and 2 these feasible dual vectors generate valid cuts. Note that a cut generated by applying (5) can be weak; the extreme case is a positive price on an infinite simple bound. In the hydroelectric scheduling problems, however, the only arcs that can have nonzero shared μ 's have natural finite bounds.

Computational Results

Columns 2-4 of Table 5 detail the performance of the base case strategy with no advanced start and columns 5-7 show the performance when we use the naive advanced start without the dual sharing formula. In selecting an advanced start procedure, one must balance the computational benefit that the preliminary cuts yield with the cost of generating the cuts. Table 5 shows the base case strategy of solving only a subset of subproblems on each stage and utilizing the dual sharing formula provides an attractive advanced start. The average relative error after the first iteration for the base case strategy and advanced start strategy with no sharing is 6.8% and 6.1%, respectively, and the corresponding average CPU times for the first iteration are 67.6 sec. and 117.8 sec. Thus the slower method produces slightly stronger cuts, but the empirical results indicate the computational expense is too high. The table reveals, however, the naive advanced start procedure is preferable to none at all, and that the advanced start procedures provide greater computational savings in the four stage problems than in the three stage problems.

Name	No Advanced Start			Naive Advanced Start		
	iter.	CPU (sec.)	% increase	iter.	CPU (sec.)	% increase
Moke3.9	15	68.4	23.5	10	60.0	8.3
Ybsf3.9	9	118.5	-1.1	7	129.1	7.8
Moke4.45	13	318.9	44.0	9	255.3	15.3
Ybsf4.45	13	614.7	52.2	9	553.1	36.9
Average	12.5	280.1	29.7	8.8	249.4	17.1

Table 5: Advanced Start Procedures

4.3 Multicut Algorithms

Birge and Louveaux [4] introduced the multicut L-shaped algorithm for SLP-2 (1). The traditional aggregate cut algorithm creates a master program by replacing $\sum_{\omega \in \Omega} p^\omega h(x_1, \omega)$ in program (1) by θ and sequentially adding cuts of the form $\theta \geq (\sum_{\omega \in \Omega} p^\omega \pi^\omega B_1) x_1 + \sum_{\omega \in \Omega} p^\omega \pi^\omega b_2^\omega$ each iteration. The multicut version instead replaces $h(x_1, \omega)$ by θ^ω for all $\omega \in \Omega$ and each iteration appends $|\Omega|$ cuts of the form $\theta^\omega \geq (\pi^\omega B_1) x_1 + \pi^\omega b_2^\omega$. When compared with the aggregate cut algorithm, the multicut algorithm has the disadvantage of requiring more decision variables; similarly, after a given number of iterations, it also maintains a larger number of cut constraints in the master program. This, however, is countered by the advantage of increased resolution of the recourse function. In practice we expect multicut algorithms will typically take fewer iterations than their aggregate cut counterparts. (Birge and Louveaux [4], however, present a counter example showing this is not always the case.) The multicut algorithm extends to the multistage setting in a straightforward fashion. Each descendant scenario passes back a cut to its ancestor and the ancestor objective function has the form:

$$c_i^{\omega_i} x_i^{\omega_i} + \sum_{\omega_{i+1} \in \Delta(\omega_i)} p_i^{\omega_{i+1}|\omega_i} \theta_i^{\omega_{i+1}|\omega_i}.$$

Other generalizations of the multicut algorithm are possible; the descendants can be partitioned into disjoint sets and a " θ " defined for each set. In the multicut algorithm described above, each set of the partition is a singleton, and the aggregate cut algorithm is the special case where the only set of the partition is $\Delta(\omega_i)$ itself. A coarse partition version of the multicut algorithm should be particularly attractive when the number of scenarios is large. The other enhancements discussed in this paper can run in either aggregate cut or multicut mode.

Computational Results

Name	# iterations	CPU (sec.)	% increase
Moke3.9	22	107.3	93.7
Ybsf3.9	13	146.3	22.1
Moke4.45	18	318.1	43.7
Ybsf4.45	17	662.0	63.9
Average	17.5	308.4	55.9

Table 6: Single Cut

Table 6 details the performance of the strategy in which the multicut method is replaced by the single cut procedure in the base case strategy. The average number of iterations in the multicut procedure is about half that of the single cut method. However, due to the quality of the warm start procedure the corresponding running times are not halved. Nevertheless, Table 6 shows the multicut method yields a significant computational advantage. The relative error as a function of CPU time is plotted in Figure 2 for three strategies on test problem Moke4.45. Note (i) the faster convergence of the multicut algorithm, (ii) the additional computational effort but improved initial relative error of the advanced start procedure, and (iii) the effect of the warm start on the time per iteration as the algorithm proceeds.

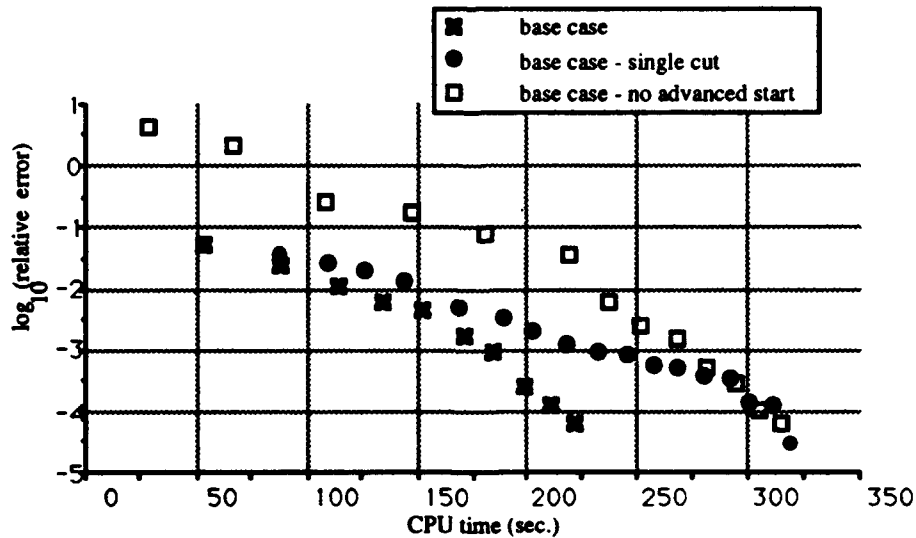


Figure 2: Relative error vs. CPU time - Moke4.45

4.4 Tree Traversing Strategies

For brevity, we refer to the nested Benders decomposition algorithm described in §3 as the *shuffle* tree traversing strategy. Abrahamson [1] and Wittrock [15] developed strategies other than *shuffle* for deterministic multistage linear programs; we consider two of these: *fastpass* and *cautious*. Gassmann [7] tested *shuffle*, *fastpass*, and *cautious* in the stochastic setting. Abrahamson, Wittrock, and Gassmann found *fastpass* to be an effective strategy. In addition to these three strategies, we present two new classes of tree traversing strategies.

The two extreme strategies are *shuffle* and *cautious*. *Shuffle* only goes backward up the tree if it cannot go forward; i.e., it solves all the stage $t + 1, \dots, T$ subproblems explicitly (within a tolerance) prior to passing cuts back to stage t . On the other hand, *cautious* never goes forward down the tree unless all cuts that would be passed back to stage $t - 1$ are redundant. *Fastpass* is a strategy “half-way” between *shuffle* and *cautious*. We introduce the ϵ -*shuffle* and ϵ -*cautious* strategies: ϵ -*shuffle* is a strategy that is in between *shuffle* and *fastpass*; it is less hesitant to go backward up the tree than the former but more hesitant than the latter; ϵ -*cautious* is similarly in between *cautious* and *fastpass*. As ϵ increases both ϵ -strategies become more like *fastpass*. As ϵ shrinks, the two ϵ -strategies more closely mimic their (non ϵ) counterparts.

The primary concern with *shuffle* is it may spend too long solving, for example, the stage $2, \dots, T$ subproblems with respect to a poor first stage decision. The quality of the information passed up the tree is high (the cuts are supports of the recourse function), but too much effort may be spent computing the cuts. *Cautious*, on the other hand, may spend too much effort generating stage $t - 1$ cuts when the stage t cuts do not yet give a good approximation of the expected costs to be incurred in stages $t + 1, \dots, T$. The “best” tree traversing strategy will properly balance the quality of the cuts (and hence the lower bound) with the computational effort required to generate them. The search for this balance motivates considering strategies that range between *shuffle* and *cautious*. *Fastpass* is one such strategy; the ϵ -strategies enable us to more fully investigate intermediate strategies.

Tree Traversing Theorem

In *shuffle*, a subproblem only receives a cut from its descendants after each descendant subproblem is solved with respect to its descendants. In this context, *solved* means the upper and lower bounds on

the optimal objective function value coincide. The tree traversing theorem states that a cut passed back to a subproblem "prematurely" (i.e., while the gaps in descendant objective function bounds are still positive) is a valid cut.

Theorem 3 *A cut passed back to $\text{sub}(\omega_t)$ when the subproblems, $\text{sub}(\omega_{t+1})$, $\omega_{t+1} \in \Delta(\omega_t)$, are not solved with respect to their descendants (due to insufficient cuts) is a valid cut.*

Proof

This result follows directly from Lemma 1 and Lemma 2: The subproblems contain valid cuts, and the dual variables associated with optimal solutions to the subproblems are dual feasible. ■

In light of Theorem 3, one has a great deal of freedom in designing algorithms with respect to the order in which the subproblems of the decision tree are solved. Our goal is to devise strategies that allow us to solve large-scale multistage stochastic linear programs as quickly as possible.

Formal Strategy Definition

Absolute error and *discrepancy* are two useful concepts in formally defining the tree traversing strategies. The absolute error, $\mathcal{A}_t(x_t^{\omega_t}, \omega_t)$, for the stage t subproblem under scenario ω_t , i.e., $\text{sub}(\omega_t)$, is

$$\mathcal{A}_t(x_t^{\omega_t}, \omega_t) = \left\{ \sum_{i=t+1}^T \sum_{\omega_i \in \Delta^{T-i}(\omega_t)} p_i^{\omega_i | \omega_t} c_i^{\omega_i} x_i^{\omega_i} \right\} - \theta_t^{\omega_t}.$$

This expression is the difference between upper and lower bounds on the optimal objective function value for $\text{sub}(\omega_t)$; note that it depends on $\text{sub}(\omega_t)$'s right-hand-side and hence its ancestor's decisions. The absolute error for the stage T subproblems is zero because these linear programs are solved directly. The Δ notation (see §2) may also be applied to a set: $\Delta(\mathcal{S}) = \bigcup_{s \in \mathcal{S}} \Delta(s)$; Δ^n means apply Δ n times, e.g., $\Delta^2(\omega_t) = \Delta(\Delta(\omega_t))$.

Scott [12] defined *discrepancy*, \mathcal{D}_t , in the deterministic case. We extend the notion of discrepancy to the stochastic setting.

$$\mathcal{D}_t(x_t^{\omega_t}, \omega_t) = \left\{ \sum_{\omega_{t+1} \in \Delta(\omega_t)} p_{t+1}^{\omega_{t+1} | \omega_t} [c_{t+1}^{\omega_{t+1}} x_{t+1}^{\omega_{t+1}} + \theta_{t+1}^{\omega_{t+1}}] \right\} - \theta_t^{\omega_t}$$

is the discrepancy for $\text{sub}(\omega_t)$. The second term in the discrepancy, $\theta_t^{\omega_t}$, represents $\text{sub}(\omega_t)$'s estimate of the expected cost to be incurred by its descendants in stages $t+1, \dots, T$. The first term represents

the conditional expected value of the cost realized in stage $t + 1$ plus the stage $t + 1$ estimate of the expected cost in stages $t + 2, \dots, T$. The discrepancy for any stage T subproblem is zero.

We now define two subroutines: *forward*(u, v) and *backward*(v, u). The former sweeps forward from stage u to stage v . It first forms the right-hand-sides of the appropriate stage u subproblems and solves them, and then it forms the right-hand-sides of the appropriate stage $u + 1$ subproblems and solves them ... until it has solved all the appropriate stage v subproblems. On the other hand, *backward*(v, u) first passes cuts to the appropriate stage v subproblems and solves them, and then passes cuts back to the appropriate stage $v - 1$ subproblems and solves them ... until it has passed cuts back to the appropriate stage u subproblems (it does not solve them). In the execution of *forward*(u, v) and *backward*(v, u), it is not always necessary to solve every subproblem that we might address. For example, in subroutine *forward*(u, v), some of the stage u subproblems (and therefore its descendants) might have a sufficiently small absolute error, and in subroutine *backward*(v, u), some of the stage v subproblems (and possibly its ancestors) might have a sufficiently small discrepancy. For declaring specific subproblems temporarily "solved" in this fashion, we use discrepancies in the *cautious* strategies and absolute errors in the *shuffle* and *fastpass* strategies.

Because the Benders cuts form an outer linearization of the recourse function, $\mathcal{D}_t(x_t^{\omega_t}, \omega_t) \geq 0$. Furthermore, $\mathcal{D}_t(x_t^{\omega_t}, \omega_t) = 0$ implies the cut that would be passed to $\text{sub}(\omega_t)$ is redundant. Two more useful facts relating absolute error and discrepancy are:

$$E_{\omega_t} \mathcal{D}_t(x_t^{\omega_t}, \omega_t) = E_{\omega_t} \mathcal{A}_t(x_t^{\omega_t}, \omega_t) - E_{\omega_{t+1}} \mathcal{A}_{t+1}(x_{t+1}^{\omega_{t+1}}, \omega_{t+1}) \quad (6)$$

$$\mathcal{A}_1(x_1^{\omega_1}, \omega_1) = \sum_{t=1}^{T-1} E_{\omega_t} \mathcal{D}_t(x_t^{\omega_t}, \omega_t). \quad (7)$$

When $\mathcal{A}_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$, SLP-T is declared to be solved where *toler* is a prespecified tolerance. U and L denote the upper and lower bounds on the optimal objective function value that the decomposition algorithm continually updates. The definitions of sufficiently small expected absolute error and sufficiently small expected discrepancy used in the tree traversing strategies are motivated by (6) and (7). The *cautious* and ϵ -*cautious* strategies begin with one iteration of *fastpass* so initial upper and lower bounds on the optimal objective function value may be defined.

(1) *fastpass*

step 0 define *toler*; set *iter* = 1
step 1 *forward*(1, *T*)
step 2 if $A_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$ then stop: optimal solution at hand
step 3 *backward*(*T* - 1, 1); *iter* = *iter* + 1
step 4 go to step 1

(2) *shuffle*

step 0 define *toler*; set *iter* = 1, *t* = 1
step 1 *forward*(*t*, *T*)
step 2 if $A_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$ then stop: optimal solution at hand
step 3 $t = \max \{i : E_{\omega_i} A_i(x_i^{\omega_i}, \omega_i) > \frac{T-1}{T-1} \cdot \text{toler} \cdot \min(|U|, |L|)\}$
step 4 *backward*(*t*, *t*)
step 5 if *t* = 1 then *iter* = *iter* + 1
step 6 go to step 1

(3) *cautious*

step 0 define *toler*; set *iter* = 1, *t*₁ = 1, *t*₂ = 2
step 1 apply one iteration of the *fastpass* algorithm; *iter* = *iter* + 1
step 2 *forward*(*t*₁, *t*₂)
step 3 if *t*₂ = *T* then
 iter = *iter* + 1;
 if $A_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$ then stop: optimal solution at hand
step 4 if $E_{\omega_{t_2-1}} \mathcal{D}_{t_2-1}(x_{t_2-1}^{\omega_{t_2-1}}, \omega_{t_2-1}) \leq \frac{\text{toler}}{T-1} \cdot \min(|U|, |L|)$ then
 *t*₁ = *t*₂ + 1; *t*₂ = *t*₁
 else
 backward(*t*₂ - 1, 1);
 *t*₁ = 1; *t*₂ = 2
step 5 go to step 2

(4) ϵ -*shuffle*

step 0 define *toler*, ϵ ; set *iter* = 1, *t* = 1
step 1 *forward*(*t*, *T*)
step 2 if $A_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$ then stop: optimal solution at hand
step 3 $t = \max \{i : \max \{j : E_{\omega_j} A_j(x_j^{\omega_j}, \omega_j) > \frac{T-1}{T-1} \cdot \epsilon \cdot \min(|U|, |L|)\}, 1\}$
step 4 *backward*(*T* - 1, *t*)
step 5 if *t* = 1 then *iter* = *iter* + 1
step 6 go to step 1

(5) ϵ -*cautious*

step 0 define *toler*, ϵ ; set *iter* = 1, *t*₁ = 1, *t*₂ = 2
step 1 apply one iteration of the *fastpass* algorithm; *iter* = *iter* + 1
step 2 *forward*(*t*₁, *t*₂)
step 3 if *t*₂ = *T* then
 iter = *iter* + 1;
 if $A_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$ then stop: optimal solution at hand
step 4 if $E_{\omega_{t_2-1}} \mathcal{D}_{t_2-1}(x_{t_2-1}^{\omega_{t_2-1}}, \omega_{t_2-1}) \leq \frac{\epsilon}{T-1} \cdot \min(|U|, |L|)$ and *t*₁ ≤ *T* - 1 then
 *t*₁ = *t*₂ + 1; *t*₂ = *t*₁
 else
 backward(*t*₂ - 1, 1);
 *t*₁ = 1; *t*₂ = 2
step 5 go to step 2

Computational Results

Table 7 restates the base case (*fastpass*) strategy's results for convenient reference. Tables 8 and 9 use "× increase" with respect to *fastpass* as the performance measure; if this ratio is less than 1.0, the strategy outperforms *fastpass* on that particular problem. By examining the # Subs columns in Tables 7 and 8 one can contrast the distribution of computing effort per stage for three strategies.

Name	<i>fastpass</i>		
	CPU (sec.)	iter.	# Subs
Moke3.9	55.4	10	10-59-99
Ybsf3.9	119.8	7	7-47-72
Moke4.45	221.4	10	10-64-274-459
Ybsf4.45	404.0	8	8-83-205-369

Table 7: Base Case Strategy - *fastpass*

Name	<i>cautious</i>			<i>shuffle</i>		
	× increase	iter.	# Subs	× increase	iter.	# Subs
Moke3.9	0.92	7	19-82-72	1.08	7	7-51-135
Ybsf3.9	1.03	6	21-86-63	1.21	6	6-45-117
Moke4.45	1.06	8	27-140-449-369	2.10	4	4-40-296-852
Ybsf4.45	1.00	6	29-202-274-279	2.18	8	8-71-263-771

Table 8: *cautious* and *shuffle*

In Table 9 the "× increase" column is the average of the CPU ratios for the four test problems. This value is displayed for some specific values of ϵ in the ϵ -*cautious* and ϵ -*shuffle* strategies.

ϵ - <i>cautious</i>		ϵ - <i>shuffle</i>	
ϵ	× increase	ϵ	× increase
0.0001	1.00	0.0001	1.64
0.0004	0.98	0.0004	1.52
0.0016	0.97	0.0016	1.32
0.0064	0.95	0.0064	1.16
0.0256	1.06	0.0256	1.08
0.1024	1.01	0.1024	0.96
0.4096	1.01	0.4096	1.00

Table 9: ϵ -*cautious* and ϵ -*shuffle*

The computational results indicate the *fastpass*, *cautious*, and ϵ -*cautious* strategies perform comparably and outperform *shuffle* and ϵ -*shuffle*. It is only as ϵ becomes large (and hence ϵ -*shuffle*

approaches *fastpass*) that ϵ -*shuffle* performs comparably. It is interesting to note how little the ϵ -*cautious* running times change as ϵ varies. An ϵ -*cautious* strategy seems appealing in a multistage problem (especially with discounting) where there is a desire to avoid the computationally expensive later stages. In these problems, however, there was not a significant difference between the three and four stage problems for the *cautious* (also ϵ -*cautious*) strategies. On the other hand, the *shuffle* (also ϵ -*shuffle*) strategy's performance is significantly worse on the four stage problems. As one might expect, the "extreme" strategies' performance deteriorates when the advanced start procedure is removed; the average CPU ratios of *cautious* and *shuffle* to *fastpass* in this case are 1.13 and 2.07, respectively.

5 Direct LP Optimizers

The performance of the enhanced decomposition algorithm (i.e., the base case) and general LP optimizers on an enlarged set of test problems is summarized in Table 10. These eight problems are based on the Mokelumne and Yuba-Bear-South Feather river basin models with 1, 9, 27, and 45 scenarios. We use the "x increase" measure with respect to the base case for three other LP solution strategies: (i) the primal-dual predictor-corrector interior point algorithm as implemented in IBM's OSL Release 2; (ii) this same interior point algorithm followed by the simplex method to generate an extreme point solution; (iii) the primal simplex method as implemented in CPLEX 2.0. The results show that on single scenario problems, the decomposition algorithm is outperformed by general LP optimizers, but as the number of scenarios grows, the decomposition algorithm is preferable.

The decomposition algorithm terminates when the relative error is less than 10^{-4} . The same duality gap tolerance was used in both interior point solution strategies; all tests were performed on a Hewlett Packard 9000/750 workstation. Due to memory limitations (OSL's *dspace* array was allocated 64 Mb) we were unable to solve the largest (Ybsf4.45) problem via the interior point strategies. Recall (§2) that the stochastic hydro scheduling problems are subproblems in a larger nonlinear Dantzig-Wolfe decomposition algorithm. Thus extreme solutions are desirable, and this is why the corresponding time to generate an optimal extreme point solution (via the simplex method) from the interior point solution is shown in Table 10.

Name	Base Case	Int. Pt.	Int. Pt. → Simplex	Simplex
	(sec.)	× increase		
Moke4.1	16.8	0.32	0.46	0.40
Moke4.9	85.0	1.00	2.41	6.31
Moke4.27	132.9	2.26	6.94	27.54
Moke4.45	221.4	3.68	11.37	47.18
Ybsf4.1	48.1	0.23	0.42	0.59
Ybsf4.9	177.3	1.12	2.94	16.04
Ybsf4.27	252.9	2.15	14.84	61.53
Ybsf4.45	404.0	N/A	N/A	100.20

Table 10: Comparison with Direct LP Optimizers

6 Summary

We have described a number of enhancements to the nested Benders decomposition algorithm for multistage stochastic linear programming. The enhanced algorithm is a small, but important part of an ongoing research and development project at The Pacific Gas and Electric Company; see Jacobs et al. [10] for a more detailed description of the project. The performance of the algorithm was examined on a collection of multistage stochastic hydroelectric scheduling problems. The computational results indicated the single most important enhancement is a warm start technique which utilizes optimal basis information from previous subproblem solutions. We also described a streamlined advanced start procedure that generates preliminary cuts to help guide the early iterations of the decomposition algorithm; this enhancement provided additional speedup over naive implementations. We found the multicut method due to Birge and Louveaux [4] also yielded computational savings over its single cut counterpart. Consistent with earlier findings of Abrahamson [1] and Wittrock [15] (in the deterministic case) and Gassmann [7] (in the stochastic case) we found that the *fastpass* tree traversing strategy performed well. However, a new class of ϵ -cautious tree traversing strategies produced comparable results to *fastpass*; further investigation of this class of strategies may be warranted for problems with many stages. Finally, we showed that taking advantage of the problem's special structure through the enhanced decomposition algorithm provides a computationally attractive alternative to direct LP optimizers on medium to large-size problems.

Acknowledgements

The author is grateful to the members of the Mid-Term Hydro-Thermal Optimization group at PG&E, headed by Konstantin Staschus, including Jonathan M. Jacobs, Jan C. Grygier, Gary L. Schultz, and Bin Zhang for their support; Zhiming Wang deserves specific thanks for valuable discussions on tree traversing strategies. This paper was written while the author was a National Research Council Research Associate at the Naval Postgraduate School, Monterey, California.

References

- [1] Abrahamson, P.G. (1983): A Nested Decomposition Approach for Solving Staircase Linear Programs, Ph.D. Dissertation, Department of Operations Research, Stanford University.
- [2] Barr, R., K. Farhangian, J.L. Kennington (1986): Networks with Side Constraints: An LU Factorization Update, *The Annals of the Society of Logistics Engineers* 1, 66-85.
- [3] Birge, J.R. (1985): Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs, *Operations Research* 33, 989-1007.
- [4] Birge, J.R. and F.V. Louveaux (1988): A Multicut Algorithm for Two-stage Stochastic Linear Programs, *European Journal of Operational Research* 34, 384-392.
- [5] Eiselt, H.A., G. Pederzoli, C.-L. Sandbloom (1987): *Continuous Optimization Models*, Berlin; New York: W. de Gruyter.
- [6] S. Garstka and D. Rutenberg (1973): Computation in Discrete Stochastic Programs with Recourse, *Operations Research* 21, 112-122.
- [7] Gassmann, H.I. (1990): MSLiP: A Computer Code for the Multistage Stochastic Linear Programming Problem, *Mathematical Programming* 47, 407-423.
- [8] Infanger, G. (1992): Planning Under Uncertainty - Solving Large-Scale Stochastic Linear Programs, Technical Report SOL 92-8, Systems Optimization Laboratory, Department of Operations Research, Stanford University.
- [9] Jacobs, J.M. (1992): Advanced Basis Insertion for Repeated Optimization of Networks with Side Constraints, Pacific Gas & Electric Company.
- [10] Jacobs, J., G. Freeman, J. Grygier, D. Morton, G. Schultz, K. Staschus, J. Stedinger, and B. Zhang (1993): Stochastic Optimal Coordination of River-basin and Thermal Electric Systems (SOCRATES): A System for Scheduling Hydroelectric Generation Under Uncertainty, submitted to *Annals of Operations Research*.
- [11] Kennington, J.L., R.V. Helgason (1980): *Algorithms for Network Programming*, John Wiley and Sons, New York, NY.
- [12] Scott, D.M. (1985): A Dynamic Programming Approach to Time-Staged Convex Programs, Technical Report SOL 85-3, Systems Optimization Laboratory, Department of Operations Research, Stanford University.
- [13] Van Slyke, R.M., R.J-B Wets (1969): L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming, *SIAM Journal of Applied Mathematics* 17, 638-663.
- [14] Wets, R. J-B (1988): Large Scale Linear Programming Techniques, in Y. Ermoliev and R.J-B Wets (eds.) *Numerical Techniques for Stochastic Optimization*, Springer Verlag, Berlin.
- [15] Wittrock, R.J. (1983): Advances in a Nested Decomposition Algorithm for Solving Staircase Linear Programs, Technical Report SOL 83-2, Systems Optimization Laboratory, Department of Operations Research, Stanford University.

INITIAL DISTRIBUTION LIST

1. Research Office (Code 08) 1
Naval Postgraduate School
Monterey, CA 93943-5000
2. Dudley Knox Library (Code 52)..... 2
Naval Postgraduate School
Monterey, CA 93943-5002
3. Defense Technical Information Center 2
Cameron Station
Alexandria, VA 22314
4. Department of Operations Research (Code OR) 1
Naval Postgraduate School
Monterey, CA 93943-5000
5. David P. Morton (Code OR/Mo)..... 40
Naval Postgraduate School
Monterey, CA 93943-5000